4. Einchipmikrorechner U881/U882-EMR

Die Grundstruktur eines Mikrorechners ist im Bild 2.1. (Seite 12) wiedergegeben. Sie besteht aus vier Basisfunktionsblöcken - der zentralen Verarbeitungseinheit (CPU), dem Programmspeicher, dem Operativdatenspeicher und aus den Ein-/Ausgabekanälen. Die zentrale Verarbeitungseinheit übernimmt die Systemsteuerung und die Befehlsabarbeitung. Der Programmspeicher enthält die Maschinenbefehle des Mikrorechners, die Schritt für Schritt von der zentralen Verarbeitungseinheit aufgerufen und abgearbeitet werden. Im Operativdatenspeicher werden Daten, die bei der Programmabarbeitung anfallen, zwischengespeichert bzw. abgelegt. Über die Ein-/Ausgabekanäle eines Mikrorechners wird die Kommunikation mit der Umwelt sichergestellt. In den behandelten U880- und U8000-Mikroprozessorsystemen (Abschnitte 2. und 3.) werden diese Funktionsblöcke durch Zusammenschaltung unterschiedlicher Bauelemente realisiert. Bei dem Einchipmikrorechner U881/U882 sind alle vier Basisfunktionen (beim U882 drei Basisfunktionen) eines Mikrorechners

zentrale Verarbeitungseinheit	(CPU)
Programmspeicher (U882 externer EPROM)	(ROM)
Operativdatenspeicher	(RAM)
Ein-/Ausgabekanäle	(PIO, UART, CTC)

auf einem einzigen Siliziumplättchen (Chip) von wenigen Quadratmillimetern Grundfläche angeordnet (Bild 4.1.) /26/,/27/,/28/,/29/.

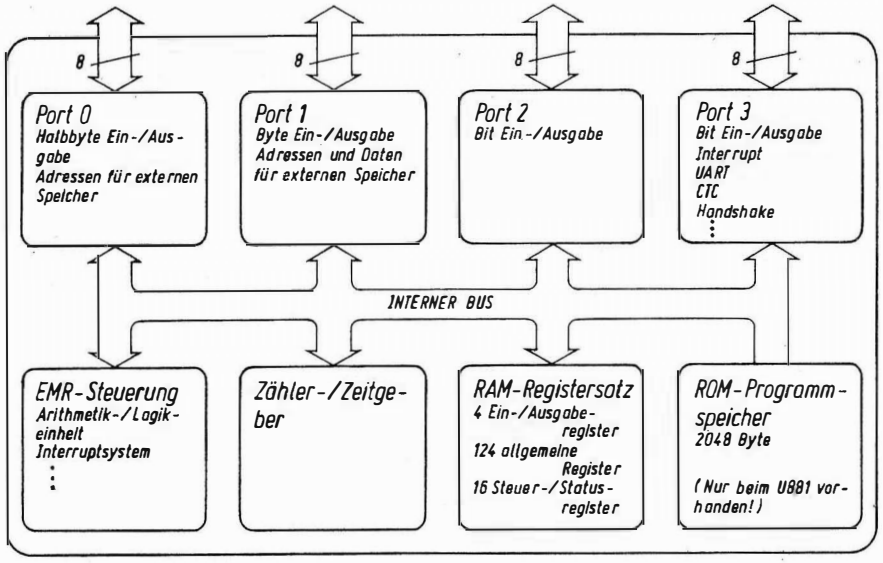


Bild 4.1. Interne Funktionskomponenten des Einchipmikrorechners U881/U882

Der U881, ein Schaltkreis mit 40 Signalanschlüssen (Pins), ist für den Einsatz in Geräten mit sehr großen Stückzahlen und unveränderlichem Programm vorgesehen. Der vom Anwender gewünschte Programmspeicherinhalt wird beim Bauelementeproduzenten /26/ fest und für alle Zeiten unveränderlich beim Herstellungsprozeß der Schaltkreise definiert (ROM-Maskenprogrammierung). Der U882 (64 Signalanschlüsse) besitzt keinen internen Programmspeicher. Die zusätzlichen Pins ermöglichen jedoch an Stelle des internen ROM den Anschluß eines externen Programmspeichers (EPROM). Die interne Struktur der beiden Schaltkreise U881 und U882 ist bis auf den beim U882 fehlenden internen Programmspeicher vollkommen identisch.
Bezogen auf die interne Struktur gilt das auch für eine dritte Variante des Einchipmikrorechners, den U883 (40 Pins), dessen interner ROM-Programmspeicher mit einem vom Anwender frei nutzbaren BASIC-Interpreter und Bootstraplader vorprogrammiert wurde.
Es ist hier anzumerken, daß der Einchipmikrorechner U881/U882 so konzipiert ist, daß er entweder als ein-/ausgabeintensiver oder als speicherintensiver Mikrorechner arbeiten kann. Das wird erreicht durch eine programmierbare Struktur der Ein-/Ausgabekanäle, die es ermöglicht, daß einige der Ein-/Ausgabekanalleitungen in einen im Multiplexverfahren betriebenen Adreßbus und Datenbus umgewandelt werden. Dadurch wird die Verwendung zusätzlicher externer Programm- und Operativdatenspeicherbereiche beim U881 und beim U882 möglich (s.a. Abschnitte 4.1.2. und 4.2.).

4.1. Zentrale Verarbeitungseinheit

4.1.1. Aufbau

Die zentrale Verarbeitungseinheit (CPU) des U881/U882 besteht im wesentlichen aus der Steuerung des Einchipmikrorechners, aus der Arithmetik-Logik-Einheit ALU (arithmetic logic unit) und aus einer ganzen Reihe von Einzelkomponenten, wie z.B. dem Vektorinterruptsystem, dem Befehlszähler PC und dem Befehlsdekoder IR (IR instruction register). Die Arithmetik-/Logik-Einheit realisiert die Befehlsausführung der 112 Basisinstruktionen. Sie ermöglicht die Behandlung von verschiedenen Datentypen (einzelne Bits in einem Byte, 4-Bit-BCD-Worte in einem Byte, Bytes und 16-Bit-Worte). Die EMR-Steuerung übernimmt die Steuerung und Überwachung des gesamten Geschehens im Einchipmikrorechner sowie die Signalgenerierung entsprechend den gegebenen Zeitabläufen für die Kommunikation mit der umgebenden Umwelt. Die interne Taktfrequenz liegt abhängig von der vorliegenden Bauelementeversion, zwischen 0,5 MHz und 4 MHz.

4.1.2. Adreßraum

Der interne Programmspeicher des Einchipmikrorechners umfaßt 2048 Byte (2 KByte). Er ist nur beim U881 aus wirklich internen, hier maskenprogrammierten ROM-Speicherzellen ausgebildet. Bei der U882-Variante des Ein-Chip-Mikroechners wird der 'interne' Speicherbereich durch einen externen Speicher, in der Regel ein EPROM-Bauelement (2K x 8 EPROM), ersetzt.
Es ist jedoch zu beachten, daß der Adreßraum des Einchipmikrorechners U881/U882 wesentlich über die 2 KByte internen Programmspeicher (ROM oder EPROM) hinausgeht. Durch entsprechende Maßnahmen kann ein externer Programmspeicherbereich von 62 KByte und zuzüglich ein ausschließlich Datenstrukturen zugeordneter externer Datenspeicherbereich von nochmals 62

KByte an den EMR-Schaltkreis über die beiden Ein-/Ausgabekanäle Port 0 und Port 1 angeschlossen werden (Bild 4.2.).

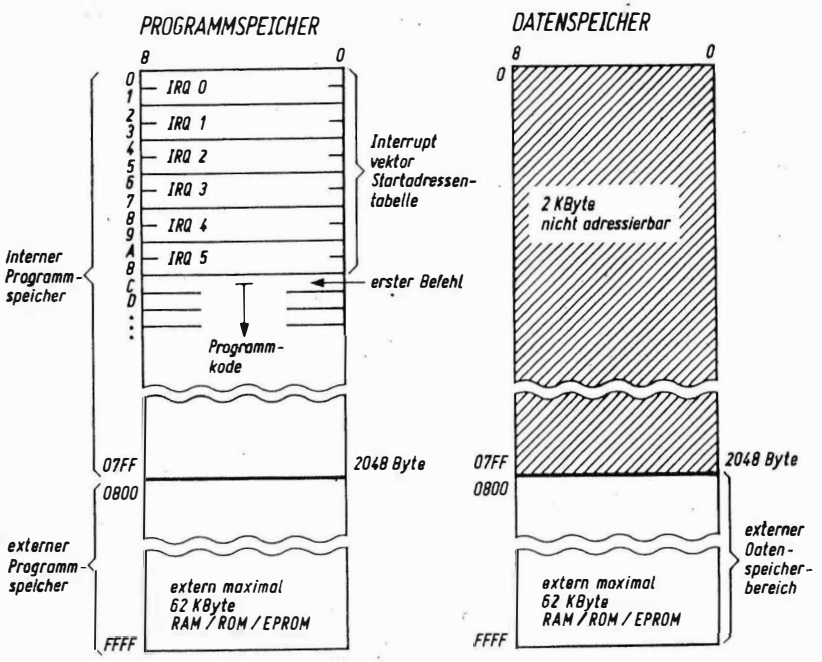


Bild 4.2. Struktur des internen und externen Programmspeichers und des externen Datenspeicherbereiches beim Einchipmikrorechner U881/U882

4.1.3. Registersatz

Der aus 144 Byte bestehende Registersatz des U881/U882 untergliedert sich in zwei Teilkomponenten, den allgemeinen Registersatz (128 Byte) und den Steuer- und Statusregistersatz (16 Byte).
Die 128 Byte des allgemeinen Registersatzes bestehen aus acht Gruppen zu je 16 Byte. Die unterste dieser Gruppen enthält auch die vier bei der Ein-/Ausgabe Verwendung findenden Ein-/Ausgabekanalregister R0, R1, R2 und R3 (Ein-/Ausgabeports) (Bild 4.3.). Sie können mit den gleichen Befehlen gelesen und geschrieben sowie arithmetisch und logisch verknüpft werden wie alle anderen Register. Spezielle Ein-/Ausgabebefehle zur Kommunikation mit der Umwelt wie beim U880 und U8000 (IN, OUT) existieren beim U881/U882 nicht. Die Einführung eines dem Programmierer zugeordneten Registerpointers (er zeigt auf eine der insgesamt neun 16 Byte Registergruppen der allgemeinen Register und der Steuer- und Statusregister) ermöglicht den

besonders schnellen und speicherplatzsparenden Zugriff innerhalb der aktuellen 16 Byte Gruppe. Diese vom Registerpointer ausgewählte Gruppe wird als Arbeitsregistergruppe bezeichnet.
Neben den Steuer- und Statusregistern existiert im U881/U882 ein 16-Bit-Befehlszählerregister PC (program counter).

4.1.4. Steuerregister/Statusregister

Die Steuer- und Statusregister (16 Byte) des U881/U882 dienen der Festlegung der Betriebsarten des Einchipmikrorechners und zur Vermittlung des EMR-Status an das laufende Programm. Das Lesen und Schreiben dieses Teiles des Registersatzes erfolgt in der gleichen Weise wie bei den allgemeinen Registern, einschließlich der Adressierungsmöglichkeit über den Registerpointer. Dabei ist zu beachten, daß ein Teil der Register (R243, R245, R246, R247, R248) nur geschrieben nicht aber gelesen werden kann (write only). Die Steuer- und Statusregistergruppe besteht aus folgenden Einzelregistern (Bild 4.3.):

Stackpointer R255(0FEH) SPL (Low-Teil) und R254(0FFH) SPH (High-Teil) - Der Stackpiointer des U881/U882 enthält, wie beim U880 und U8000, die Adresse des im Speicher angeordneten Stackspeicherbereichs. Er dient zur Aufnahme der Rückkehradressen bei Unterprogrammaufrufen und Interruptserviceroutinen, kann allerdings mit Hilfe der Befehle PUSH und POP auch zur Aufbewahrung verschiedener bei der Programmabarbeitung anfallender Operativdaten, zur Parameterübergabe bei Unterprogrammen u.a.m. benutzt werden. Beim U881/U882 kann entweder mit einem internen Stack (innerhalb des allgemeinen Registersatzes) oder mit einem externen Stack (im externen Datenspeicherbereich) gearbeitet werden. Dementsprechend ist ein 8-Bit-Stackpointer (nur R255) ausreichend (R254 ist dann frei verfügbar) oder es muß ein 16-Bit-Stackpointer (R255 und R254) benutzt werden. Die Festlegung, ob ein interner oder ein externer Stack verwendet wird, erfolgt über das Datenbit D2 im Register R248 (Port 0 / Port 1 Moderegister).

Registerpiointer R253(0FDH) RP - Der Registerpointer dient zur Auswahl einer als Arbeitsregister bezeichneten Gruppe von 16 Einzelregistern aus dem Gesamtregistersatz des Einchipmikrorechners. Der Inhalt der oberen vier Bit des Registerpointers zeigt bei diesem Arbeitsregisteradressierungsverfahren auf das erste Register jeweils einer Befehlsgruppe. Die unteren vier Bit werden als interne 4-Bit-Registeradresse dem aktuellen Befehlskode entnommen. Zusammen entsteht aus zwei 4-Bit-Gruppen eine 8-Bit-Adresse mit 256 Einzeladressen, genau den Registeradreßraum der U881/U882 überstreichend. Solange sich der Programmierer innerhalb einer 16 Byte umfassenden Arbeitsregistergruppe bewegt, bleibt der Registerpointer unverändert und es kann mit den ein Byte kürzeren, nur auf die jeweilige Arbeitsregistergruppe bezogenen Befehlen gearbeitet werden. Der Übergang von einer Arbeitsregistergruppe in eine andere erfolgt durch Umladen des Registerpointers (Bild 4.3.).

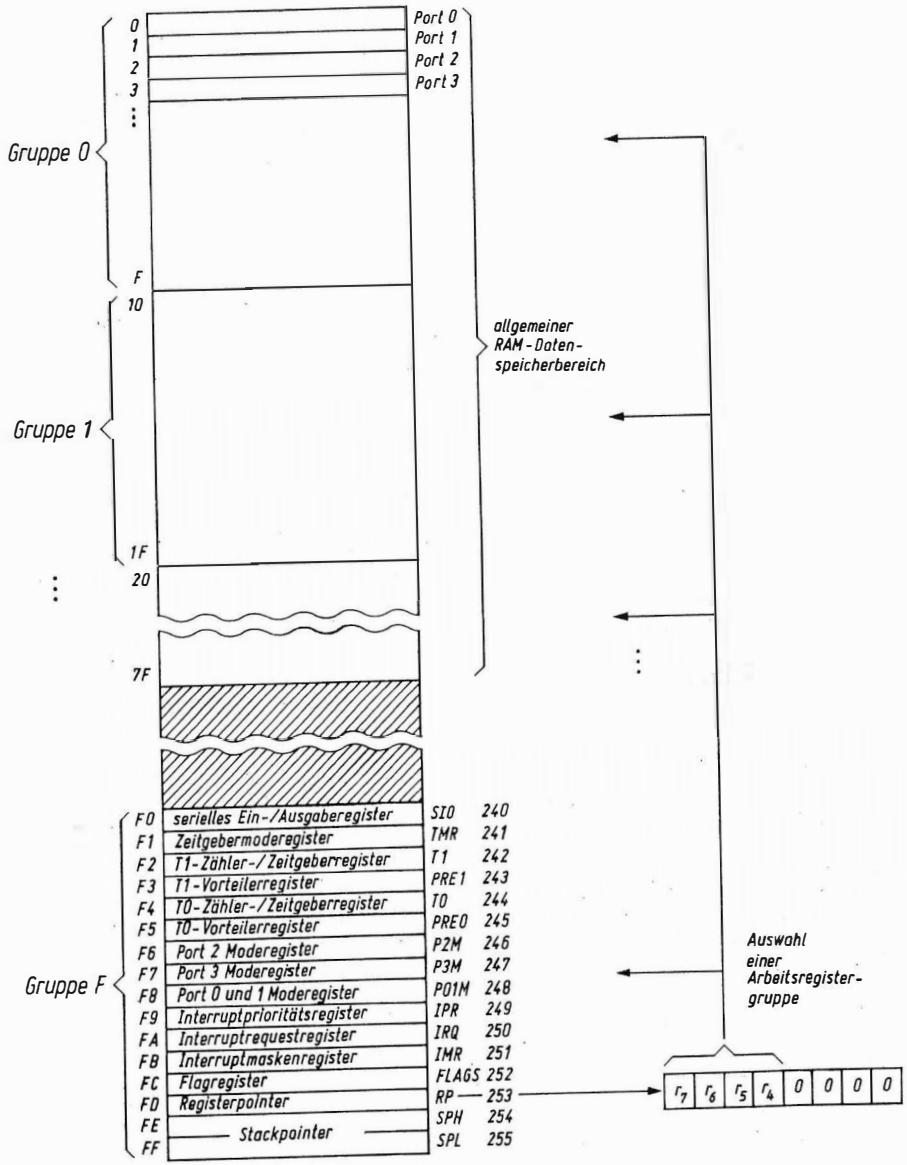


Bild 4.3. Aufbau des U881/U882-EMR Registersatzes

Flagregister R252(OFCH) FLAGS - Der Inhalt des Registers R252 gibt Auskunft über verschiedene im Ergebnis der Befehlsausführung im U881/U882 eingetretene Statusmodifikationen und dient zur bedingungsabhängigen Gestaltung des Programmablaufes. Die Funktionen der Flagbits C, Z, S, V, D und H entsprechen bei der EMR-Befehlsabarbeitung weitgehend der Funktion der äquivalenten U880-Flags (s.a. Abschn. 2.1.3. sowie EMR-Befehlsliste Tafel 4.1.). Zwei Flags F1 und F2 sind vollkommen frei vom Programmierer verwendbar (alle Bits des Flagregisters können sowohl gelesen als auch geschrieben werden).
Im einzelnen hat das Flagregister folgende Belegung:

C	Z	S	V	D	H	F2	F1

C-Flag Übertragsflag (carry)
Z-Flag Nullflag (zero)
S-Flag Vorzeichenflag (sign)
V-Flag Überlaufsflag (overflow)
D-Flag BCD-Normalisierungsflag (decimal adjust)
H-Flag Halbübertragsflag (half carry)
F2-Flag Anwenderflag 2 (user flag 2)
F1-Flag Anwenderflag 1 (user flag 1)

Interruptmaskenregister R251(OFBH) IMR - Das Interruptmaskenregister ermöglicht dem Programmierer, das Interruptsystem des U881/U882 insgesamt oder selektiv (sechs Einzelinterruptquellen) ein- bzw. auszuschalten. Dem Ein- und Ausschalten des gesamten EMR-Interruptsystems ist das Datenbit D7 zugeordnet. Es wird mit einem speziellen Befehl gesetzt EI (=1 / enable interrupt) und rückgesetzt DI (=0 / disable interrupt). Gesetzt wird dieses Flagbit außerdem automatisch bei der Abarbeitung des IRET-Befehls (return from interrupt) und rückgesetzt im Interruptanerkennungszyklus.

EI	0/1	IRQ5	IRQ4	IRQ3	IRQ2	IRQ1	IRQ0

Zu beachten ist, daß das Datenbit D7 im Interruptmaskenregister nach dem Einschalten des U881/U882 (Systemreset) mit dem DI-Befehl rückgesetzt werden muß, bevor es selbst und das in der Folge noch zu beschreibende Interruptprioritätsregister erstmalig beschrieben werden kann. Die Datenbits D0 bis D5 des Interruptmaskenregisters sind den sechs EMR-Interruptquellen IRQ0 bis IRQ5 zugeordnet. Werden die entsprechenden Datenbits gesetzt (=1), sind die korrespondierenden Einzelinterruptsignale zugelassen - werden sie rückgesetzt (=0), sind sie gesperrt. (D6 im IMR wird nicht benutzt.)

Interruptrequestregister R250(OFAH) IRQ - Das Interruptrequestregister speichert eine anstehende Interruptanforderung bis zu ihrer Abarbeitung. Trifft eine Einzelinterruptanforderung ein, wird das korrespondierende Datenbit D0...D5 entsprechend IRQ0...IRQ5 auf den Wert Eins gesetzt. Nach

seiner Akzeptierung im Interruptanerkennungszyklus (interrupt machine cycle) wird es automatisch rückgesetzt.

0/1	0/1	IRQ5	IRQ4	IRQ3	IRQ2	IRQ1	IRQ0

Da es sich beim IRQ-Register um ein Schreib-/Leseregister handelt, in dem auch vom Programmierer einzelne Datenbits gesetzt und rückgesetzt werden können, ergibt sich, daß durch Lesen dieses Registers ermittelt werden kann, ob einzelne Interruptanforderungen eingegangen sind oder nicht (polling). Durch Schreiben des IRQ-Registers kann der Programmierer Einzelinterruptsignale simulieren, in dem er das entsprechende Datenbit im Interruptrequestregister auf Eins setzt. (D7 und D6 im ITQ werden nicht benutzt.)

Interruptprioritätsregister R249(0F9H) IPR - Das Interruptprioritätsregister hat die Aufgabe, bei gleichzeitigem Eintreffen oder bei gleichzeitigem Anstehen (nach einer Interruptsperre) von mehreren Einzelinterruptsignalen zu entscheiden, welche der verschiedenen Interruptanforderungen zuerst bedient werden soll. Mit Hilfe dieses Registers können die sechs Einzelinterrupts in 48 verschiedene Prioritätsverteilungsarten aufgelöst werden. Die Interruptprioritätsermittlung über das Register R249 ist nur bei gleichzeitigem Anstehen mehrerer Interruptanforderungen aktiv, die prioritätsgesteuerte Verschachtelung (nesting) der Interruptbedienung wie beim U880 und U8000 über eine Prioritätskaskade (daisy chain) wird vom U881/U882 hardwaremäßig nicht unterstützt. (D7 und D6 im IPR werden nicht benutzt.)
Prioritätsfestlegung im IPR (A, B und C sind Prioritätsgruppen):

```
D4,D3,D0=000  ---                      Gruppe A:  D1=0  IRQ1 größer IRQ4
        =001  C größer A größer B                 =1   IRQ4 größer IRQ1
        =010  A größer B größer C      Gruppe B:  D2=0  IRQ2 größer IRQ0
        =011  A größer C größer B                 =1   IRQ0 größer IRQ2
        =100  B größer C größer A      Gruppe C:  D5=0  IRQ5 größer IRQ3
        =101  C größer B größer A                 =1   IRQ3 größer IRQ5
        =110  B größer A größer C
        =111  ----
```

(Steuer- und Statusregister R240, R241 ...R248 s. Abschnitt 4.2. Ein-/Ausgabe.)

4.1.5. Interruptsystem

Das Vektorinterruptsystem des U881/U882 erlaubt die Behandlung von sechs verschiedenen Interrupts aus acht unterschiedlichen Quellen (Bilder 4.4. und 4.5.).
Vier Interruptquellen (IRQ0...IRQ3) sind den vier Eingangssignalen des Ein-/Ausgabeports 3 zugeordnet. Sie können wahlweise den Eingangsbits dieses Ports (Flanke 'high' auf 'low'), den Quittungsbetriebssignalen (handshake) DAV0/RDY0, DAV1/RDY1 und DAV2/RDY2 vom Port 0, 1 oder 2 oder aber dem internen CTC- bzw. dem USART-Eingangssignal des Einchipmikrorechners zugeordnet werden. Die beiden Interruptquellen IRQ4 und IRQ5

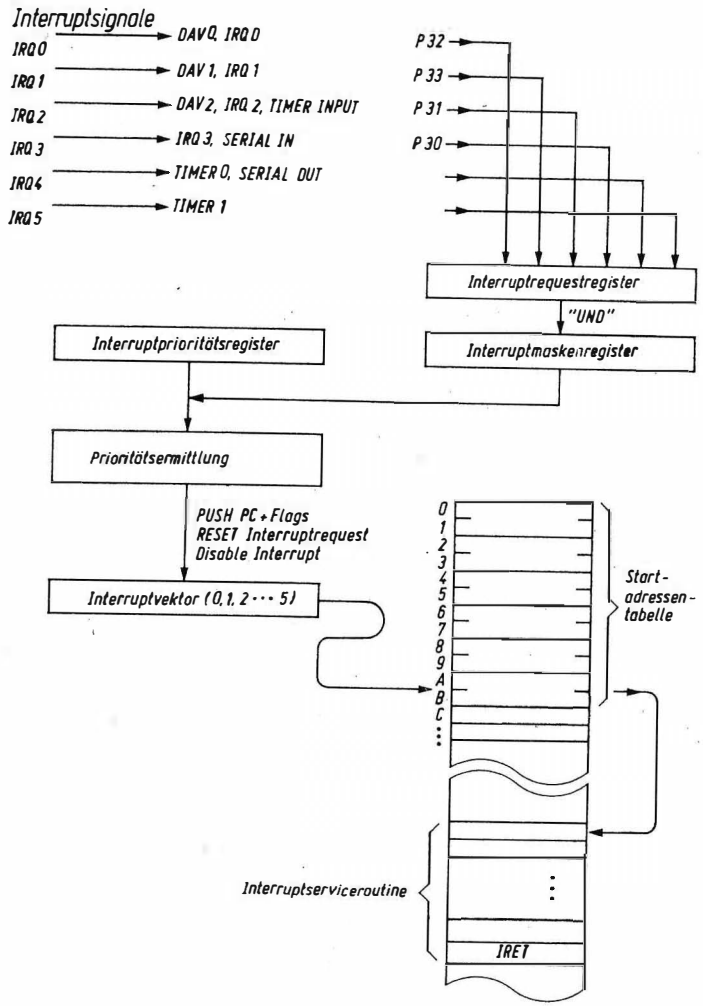


Bild 4.4. Struktur des U881/U882 Vektorinterruptsystems

entsprechen wahlweise den beiden CTC-Ausgangssignalen (TO und T1) oder dem CTC-Ausgangssignal (T1) und dem USART-Ausgangssignal (USART universal asynchronous receiver transmitter / s. Abschnitt 4.2.). Die Programmierung welche Interruptquelle für ein bestimmtes Bit anzusetzen ist, erfolgt mit der Programmierung des Ein-/Ausgabeports 3 (Register R247) und mit der Programmierung der beiden CTC-Kanäle (Register R241, R242, R243 und R244). Der Ablauf einer Interruptbehandlung beim U881/U882 ist dem Bild 4.4. zu entnehmen, dabei ist der Einfluß der Bitbelegung des Interruptmaskenregisters (R249), des Interruptrequestregisters (R250) und des Interruptprioritätsregisters (R249) auf die Interruptbearbeitung zu beachten.
Zu beachten ist außerdem, daß bei einem Interrupt (ähnlich wie beim U8000 aber im Gegensatz zum U880) nicht nur der Befehlszählerinhalt, sondern auch der Inhalt des Flagregisters R252 automatisch in den Stack eingespeichert wird - zusammen drei Byte (s.a. IRET-Befehl Tafel 4.1.).

4.1.6. Adressierungsarten

Der Befehlssatz des Einchipmikrorechners U881/U882 beinhaltet fünf Adressierungsarten:

Registeradressierung	Der Operand ist der Inhalt eines Registers.
Indirekte Registeradressierung	Der Operand ist der Inhalt einer Adresse, die durch den Inhalt eines Registers ausgewählt wird.
Direktwert Adressierung	Der Operand ist Teil des Befehls.
Indizierte Adressierung	Der Operand ist der Inhalt einer Adresse, deren Wert sich aus dem Inhalt eines Registers, addiert um den Inhalt eines im Befehl angegebenen Arbeitsregisters (Index), ergibt.
Relative Adressierung	Der Operand ist der Befehlszählerinhalt, versetzt (plus/minus) um den im Befehl angegebenen Distanzwert (displacement).

4.1.7. Befehlssatz

Der Befehlssatz des Einchipmikrorechners U881/U882 umfaßt die Befehlsgruppen:

Ladebefehle,
Blocktransferbefehle,
Arithmetik-/Logikbefehle,
Rotations-/Verschiebebefehle,
Sprung-/Call-/Returnbefehle und
EMR-Steuerbefehle.

Die U881/U882-Befehlstypen ähneln den U880- und den U8000-Befehlstypen in vieler Hinsicht.
Die Befehle der Ladebefehlsgruppe transportieren Daten zwischen den EMR-Registern entsprechend den bei diesem Schaltkreis vorhandenen Adressierungsverfahren. Sie dienen dem Laden von Konstanten (immediate) in

die Register, dem Löschen von Registern und dem Ein- und Ausspeichern von Registerinhalten in den Stackspeicherbereich. Vier Befehle dieser Gruppe ermöglichen den Datentransport zwischen den EMR-Registern sowie dem externen Programm- und Datenspeicherbereich. Die Blocktransferbefehle (LDCI/LDEI) dienen dem Datentransfer von und zum Programm- und Datenspeicher mit automatischer Inkrementierung von zwei Zählern.
Die Arithmetik-/Logikbefehle bieten dem Programmierer die vom U880 und U8000 in ihrer Wirkungsweise weitgehend bekannten Standardbefehle zur Addition, zur Subtraktion, zum Inkrementieren/Dekrementieren (hier 8- und 16-Bit-Wörter), zum Vergleich sowie für die logischen Operationen AND, OR, XOR und die logische Negation an. Alle Befehle arbeiten mit 8-Bit-Operanden in einem EMR-Register. Ein Testbefehl für Bitwerte mit Hilfe einer Maske ergänzt diese Gruppe. Zu beachten ist hier die durchgängige Beeinflussung der Flagbits.
Die Rotations- und Verschiebebefehle gestatten die bitweise Rotation und Verschiebung von Registerinhalten.
Die Sprung-, Call- und Returnbefehle dienen dem Aufbau von Programmverzweigungen, von Programmschleifen, der Unterprogrammorganisation und der Rückkehr aus Interruptserviceroutinen. Die EMR-Steuerbefehle schließlich ermöglichen die Manipulation des C-Flagbit, des U881/U882-Interruptsystems und des Registerpointers.
Die Effektivität der Befehle sowohl in bezug auf die Abarbeitungsgeschwindigkeit (typisch 2,5 µs für ein 4 MHz U881/U882-System) als auch in bezug auf Befehlskodelänge (Arbeitsregisteradressierungsschema) ist sehr groß, so daß mit den im konkreten Einsatzfall meist begrenzten EMR-Ressourcen doch recht komplexe Aufgaben gelöst werden können. Durch den überlappenden Befehlsaufruf (instruction pipelining), bei dem sich die Abarbeitung des vorhergehenden Befehls mit dem Befehlslesen des nächsten teilweise überdeckt, wird zusätzlich eine Optimierung der Programmabarbeitungszeit erreicht.

Tafel 4.1. Befehlsliste des Einchipmikrorechenrs U881/U882

ZEICHENERLÄUTERUNG

r,r'	Arbeitsregister (Rn n=0...15)
R,R'	Register (reg 0...127,240...255)
RR	Registerpaar (reg 0,2...126,240,242...254)
Ir	Arbeitsregister indirekt (§Rn n=0...15)
IR	Register indirekt (§reg 0...127,240...255)
Irr	Arbeitsregisterpaar indirekt (§RRp p=0,2,4...14)
IRR	Registerpaar indirekt (§reg 0,2,4...126,240,242...254)
DA	direkte Adresse (16-Bit-Konstante)
RA	relative Adresse (8-Bit-Konstante)
IM	Direktwert (8-Bit-Konstante)

%	Hexadezimalkennzeichen
§	Indirektkennzeichen
*	Befehlszählerinhalt

Flagbitbedingungskode (cc):

	FALSE (falsch)	0000	----
	TRUE (richtig)	1000	----
Z	ZERO	0110	Z=1
NZ	NOT ZERO	1110	Z=0
C	CARRY	0111	C=1
NC	NOT CARRY	1111	C=0
PL	PLUS	1101	S=0
MI	MINUS	0101	S=1
NE	NOT EQUAL	0110	Z=0
EQ	EQUAL	1110	Z=1
OV	OVERFLOW	0100	V=1
NOV	NO OVERFLOW	1100	V=0
GE	GREATER THAN OR EQUAL	1001	S XOR V=0
LT	LESS THAN	0001	S XOR V=1
GT	GREATER THAN	1010	Z OR (S XOR V)=0
LE	LESS THAN OR EQUAL	0010	Z OR (S XOR V)=1
UGE	UNSIGNED GE	1111	C=0
ULT	UNSIGNED LT	0111	C=1
UGT	UNSIGNED GT	1011	(C=0) AND (Z=0)=1
ULE	UNSIGNED LE	0011	C OR Z=1

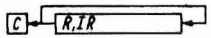
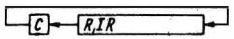
Flagbitbeeinflussung:
. unverändert
x unbestimmt
? gesetzt (0 oder 1)
0 Null gesetzt
1 Eins gesetzt

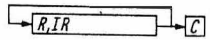
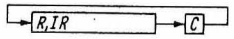
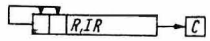
T1/T2 Befehlsausführungszyklen/Befehlspipelinezyklen (0,4 - 1,0 µs)

LADEBEFEHLE

Befehl	T1	T2	Kodierung	Erläuterung	C Z S V D H
LD r,R	6	5	--r-1000 ----R---	r:=R	. . . . . .
LD R,r	6	5	--r-1001 ----R---	R:=r	. . . . . .
LD r,Ir	6	5	11100011 --r--Ir-	r:=Ir	. . . . . .
LD Ir,r	6	5	11110011 -Ir---r-	Ir:=r	. . . . . .
LD R,R'	10	5	11100100 ----R'-- ----R---	R:=R'	. . . . . .
LD R,IR	10	5	11100101 --IR--- ----R---	R:=IR	. . . . . .
LD IR,R	10	5	11110101 ----R--- ---IR---	IR:=R	. . . . . .

Befehl	T1	T2	Kodierung	Erläuterung	C Z S V D H
LD r,IM	6	5	--r-1100 ---IM---	r:=IM	. . . . . .
LD R,IM	10	5	11100110 ----R--- ---IM---	R:=IM	. . . . . .
LD IR,IM	10	5	11100111 ---IR--- ---IM---	IR:=IM	. . . . . .
LD r,X	10	5	11000111 --r--r'- ----R---	r:=X indiziert X=reg(Rn)	. . . . . .
LD X,r	10	5	11010111 --r--r'- ----R---	X:=r indiziert X=reg(Rn)	. . . . . .
CLR R CLR IR	6 6	5 5	10110000 10110001 ---IR---	R:=0 IR:=0	. . . .
LDC r,Irr	12	0	11000010 --r-Irr-	r:=Irr Laden aus (in) dem (den)	. . . . . .
LDC Irr,r	12	0	11010010 Irr---r-	Irr:=r Programmspeicher	. . . . . .
LDE r,Irr	12	0	10000010 --r-Irr-	r:=Irr Laden aus (in) dem (den)	. . . . . .
LDE Irr,r	12	0	10010010 Irr---r-	Irr:=r Datenspeicher	. . . . . .
PUSH R	10 12	1	01110000 ----R---	SP:=SP-1 (10 int.) §SP:=R (12 ext.)	. . . . . .
PUSH IR	12 14	1	01110001 ---IR---	SP:=SP-1 (12 int.) §SP:=IR (14 ext.)	. . . . . .
POP R	10	5	01010000 ----R---	R:=§SP SP:=SP+1	. . . . . .
POP IR	10	5	01010001 ---IR---	IR:§SP SP:=SP+1	. . . . . .
BLOCKTRANSFERBEFEHLE					
LDCI Ir,Irr	18	0	11000011 -Ir--Irr	Ir:=Irr Transfer aus r:=r+1 dem Programm- rr:rr+1 speicher	. . . . . .
LDCI Irr,Ir	18	0	11010011 -Irr-Ir-	Irr:=Ir Transfer in r:=r+1 den Programm- rr:=rr+1 speicher	. . . . . .
LDEI Ir,Irr	18	0	10000011 -Ir--Irr	Ir:=Irr Transfer aus r:=r+1 dem Daten- rr:=rr+1 speicher	. . . . . .
LDEI Irr,Ir	18	0	10010011 -Irr-Ir-	Irr:=Ir Transfer in r:=r+1 den Daten- rr:=rr+1 speicher	. . . . . .
ARITHMETIK-/LOGIKBEFEHLE					
ADD r,r'	6	5	00000010 --r--r'-	r:=r+r'	? ? ? ? 0 ?
ADD r,Ir	6	5	00000011 --r--Ir-	r:=r+Ir	? ? ? ? 0 ?
ADD R,R'	10	5	00000100 ----R'-- ----R---	R:=R+R'	? ? ? ? 0 ?

ADD R,IR	10	5	00000101 ---IR--- ----R---	R:=R+IR		? ? ? ? 0 ?
ADD R,IM	10	5	00000110 ----R--- ---IM---	R:=R+IM		? ? ? ? 0 ?
ADD IR,IM	10	5	00000111 ---IR--- ---IM---	IR:=IR+IM		? ? ? ? 0 ?
ADC s,s' SUB s,s' SBC s,s' OR s,s' AND s,s' XOR s,s' CP s,s' TCM s,s' TM s,s'			0001 0010 0011 0100 0101 1011 1010 0110 0111	s:=s+s'+C s:=s-s' s:=s-s'-C s:=s OR s' s:=s AND s' s:=s XOR s' s=s' ? TEST s-Bit 0 (s neg UND s') TEST s-Bit 1 (s UND s')	s kann sein: r,Ir, R,IR oder IM Kodierung analog ADD 0000 wird substituiert	? ? ? ? 0 ? ? ? ? ? 1 ? ? ? ? ? 1 ? . ? ? 0 . . . ? ? 0 . . . ? ? 0 . . ? ? ? ? . . . ? ? 0 . . . ? ? 0 . .
COM R	6	5	01100000 ----R---	R:=R logisch negiert		. ? ? 0 . .
COM IR	6	5	01100001 ---IR---	IR:=IR logisch negiert		. ? ? 0 . .
DA R	8	5	01000000 ----R---	DEZIMAL ADJUST R (BCD)		? ? ? x . .
DA IR	8	5	01000001 ---IR---	DEZIMAL ADJUST IR (BCD)		? ? ? x . .
INC r	6	5	--r-1110	r:=r+1 8 Bit		. ? ? ? . .
INC R	6	5	00100000 ----R---	R:=R+1 8 Bit		. ? ? ? . .
INC IR	6	5	00100001 ---IR---	IR:=IR+1 8 Bit		. ? ? ? . .
DEC R	6	5	00000000 ----R---	R:=R-1 8 Bit		. ? ? ? . .
DEC IR	6	5	00000001 ---IR---	IR:=IR-1 8 Bit		. ? ? ? . .
INCW RR	10	5	10100000 ---RR---	RR:=RR+1 16 Bit		. ? ? ? . .
INCW IR	10	5	10100001 ---IR---	IR:=IR+1 16 Bit		. ? ? ? . .
DECW RR	10	5	10000000 ---RR---	RR:=RR-1 16 Bit		. ? ? ? . .
DECW IR	10	5	10000001 ---IR---	IR:=IR-1 16 Bit		. ? ? ? . .
ROTATIONS-/VERSCHIEBEBEFEHLE						
RL R	6	5	10010000 ----R---			? ? ? ? . .
RL IR	6	5	10010001 ---IR---			? ? ? ? . .
RLC R	6	5	00010000 ----R---			? ? ? ? . .
RLC IR	6	5	000100001 ---IR---			? ? ? ? . .

Befehl	T1	T2	Kodierung	Erläuterung	C Z S V O H
RR R	6	5	11100000 ----R---		? ? ? ? . .
RR IR	6	5	11100001 ---IR---		? ? ? ? . . ? ? ? ? . .
RRC R	6	5	11000000 ----R---		? ? ? ? . .
RRC IR	6	5	11000001 ---IR---		? ? ? ? . .
SRA	6	5	11010000 ----R---		? ? ? 0 . .
SRA IR	6	5	11010001 ---IR---		? ? ? 0 . .
SWAP R	8	5	11110000 ----R---		x ? ? x . .
SWAP IR	8 8	5 5	11110001 ---IR---		x ? ? x . .
SPRUNG-/CALL-/RETURNBEFEHLE					
JP IRR	8	0	00110000 --IRR---	PC:=IRR	. . . . .
JP cc,DA	12 10	0 0	-cc-1101 ---DA_H-- ---DA_L---	PC:=DA cc TRUE PC:=PC+3 cc FALSE	. . . . .
JR cc,RA	12 10	3 0	-cc-1011 ---RA---	PC:=PC+RA cc TRUE PC:=PC+2 cc FALSE	. . . . .
DJNZ RA	12	3	--r-1010 ---RA---	r:=r+1 wenn r≠0 PC:=PC+RA wenn r=0 PC:=PC+2	. . . . .
CALL DA	20	0	11010110 ---DA_H-- ---DA_L--	SP:=SP-2 §SP:=PC+3 und PC:=DA	. . . . .
CALL IRR	20	0	11010100 --IRR---	SP:=SP-2 §SP:=PC+2 und PC:=IRR	. . . . .
RET	14	0	10101111	PC:=§SP SP:=SP+2	. . . . .
IRET	16	0	10111111	FLAGS:=§SP PC:=§SP SP:=SP+3 und IMR_7:=1	? ? ? ? ? ?
EMR-STEUERBEFEHLE					
CCF	6	5	11101111	C:=C negiert	? . . . . .
RCF	6	5	11001111	C:=0	0 . . . . .
SCF	6	5	11011111	C:=1	1 . . . . .
DI	6	1	10001111	IMR_7:=0	. . . . .
EI	6	1	10011111	IMR_7:=1	. . . . .
SRP IM	6	1	00110001 ---IM---	RP:=IM (Registerpointer)	. . . . .
NOP	6	0	11111111	no operation	. . . . .

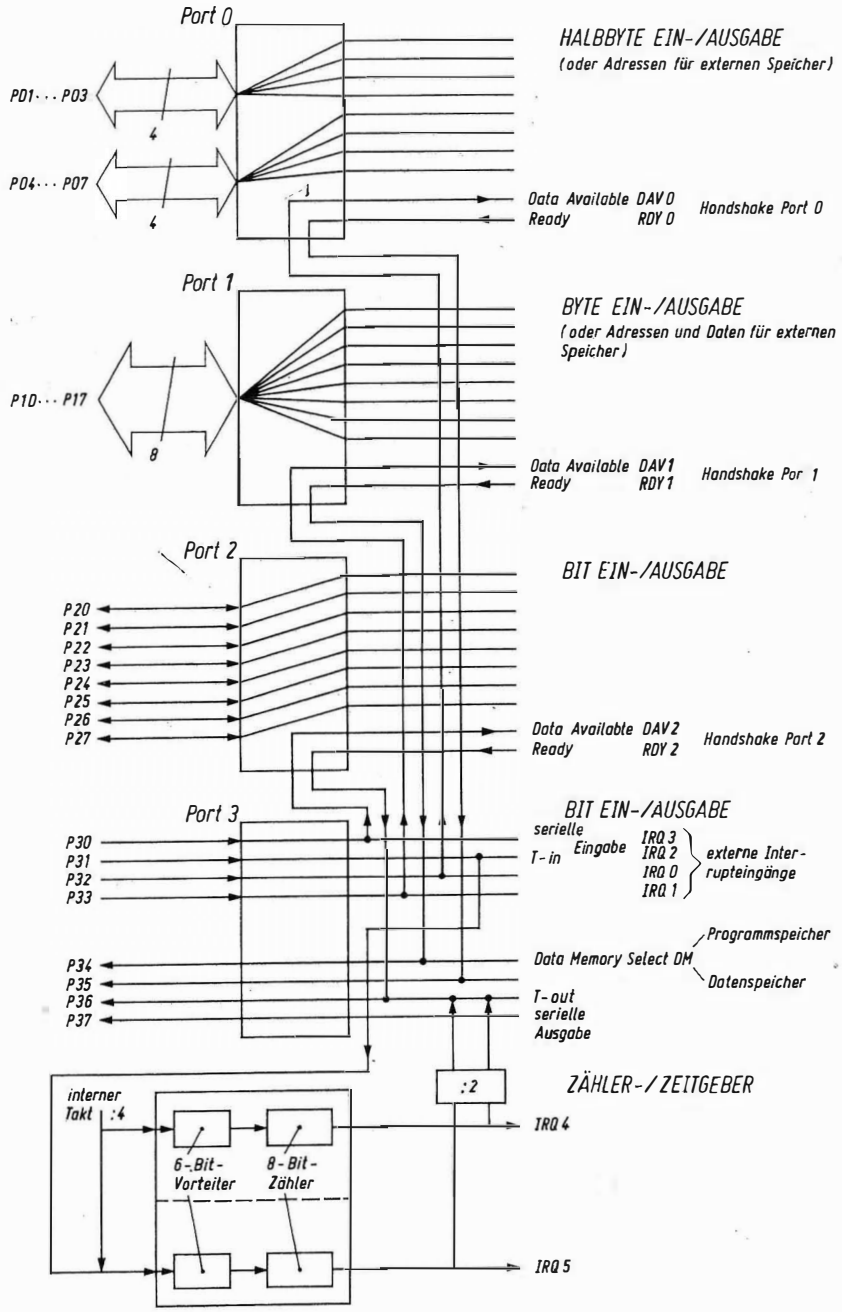


Bild 4.5. Ein-/Ausgabekanäle des Einchipmikrorechners U881/U882

4.2. Ein-/Ausgabekanäle

4.2.1. Aufbau

Insgesamt 32 Signalanschlüsse des Einchipmikrorechners U881/U882 werden zur Datenkommunikation mit der Umwelt verwendet (Bild 4.5.). Sie können zur parallelen Ein-/Ausgabe (Port 0, 1, 2 und 3), zur seriellen Ein-/Ausgabe (Port 3 Bit P30 serial in - P37 serial out), für Zähler-/Zeitgeberaktivitäten (Port 3 Bit P31 T-in - P36 T-out) oder auch zur Ankopplung externer Programm- und Datenspeicherbereiche (Port 0 und 1) genutzt werden.
Durch die spezielle Programmierung der U881/U882-Ein-/Ausgabekanäle stehen dem Anwender hier außerordentlich vielfältige Variationsmöglichkeiten offen. Die Ein- und Ausgabe von Daten über die vier E/A-Ports erfolgt durch Lesen (Eingabe) bzw. Schreiben (Ausgabe) der korrespondierenden Register R0, R1, R2 oder R3. Spezielle Ein-/Ausgabebefehle existieren beim U881/U882-EMR nicht.

Ein-/Ausgabeport 0 - Der Ein-/Ausgabeport 0 dient zur Eingabe oder Ausgabe von zwei 4-Bit-Datenströmen (Halbbyte Ein-/Ausgabe / nibble input output). Außerdem laufen gegebenenfalls über den Port 0, ebenso wie über den Port 1, je nach Programmierung vier oder acht Adreßsignalleitungen (A8...A11 oder A8...A15) bei Verwendung eines externen Programm- und (oder) Datenspeichers. Ein für die Übertragung von Adreßsignalen verwendetes Halbbyte dieses Datenkanals kann nicht mehr (oder nur durch externe Zusatzmaßnahmen und spezielle Software) für die Ein-/Ausgabe verwendet werden. Wird der Ein-/Ausgabeport 0 teilweise oder ausschließlich für den Ein-/Ausgabetransfer verwendet, können bei Bedarf über die Datenbits P32 und P35 des Ein-/Ausgabekanals 3 zwei Handshakesignalleitungen DAV0 und RDY0 (DAV data available / RDY ready) aktiviert werden (Bild 4.6.).

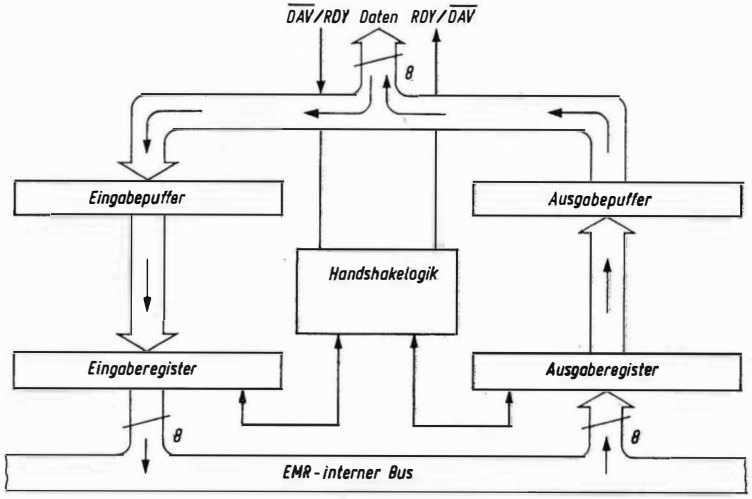


Bild 4.6. Interner Aufbau der Ein-/Ausgabekanäle 0, 1, und 2 des U881/U882

Ein-/Ausgabeport 1 - Der Ein-/Ausgabeport 1 dient zur Eingabe oder Ausgabe eines 8-Bit-Datenstromes (Byte Ein-/Ausgabe). Außerdem laufen gegebenenfalls über den Port 1 - ähnlich wie über den Port 0 - acht Adreßsignal- und Datensignalleitungen (AD0...AD7 / multiplex) bei Verwendung eines externen Programm- und (oder) Datenspeichers. Wird nur der Ein-/Ausgabeport 1 zum Anschluß externer Speicherbereiche benutzt, können maximal 256 Adressen (8 Adreßleitungen) angesprochen werden. Wird ein größerer Adreßbereich benötigt muß zusätzlich der Port 0 zur Adreßsignalvermittlung benutzt werden.
Auch dem Port 1 können bei Bedarf zwei Handshakesignalleitungen DAV1 und RDY1 analog denen des Ein-/Ausgabekanals 0 zugeordnet werden (Datenbits P33 und P34).

Ein-/Ausgabeport 2 - Der Ein-/Ausgabeport 2 dient zur Eingabe und Ausgabe von acht Einzelsignalen, die bitweise auf Eingabe oder Ausgabe geschaltet werden können.
Für die gegebenenfalls notwendige Handshakesteuerung - DAV2 und RDY2 - des Ein-/Ausgabeverkehrs gilt das beim Port 0 und 1 Gesagte (Datenbits P31 und P36).

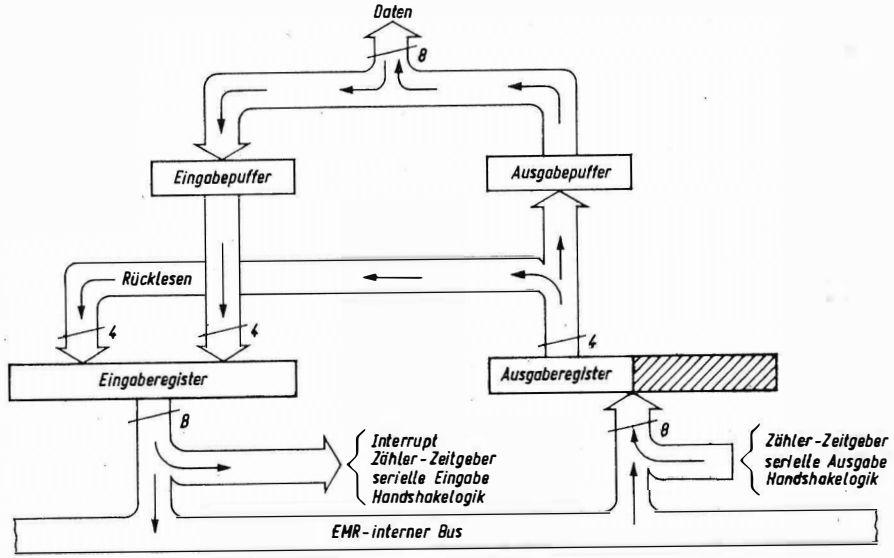


Bild 4.7. Interner Aufbau des Ein-/Ausgabekanals 3 des U8B1/U882

Ein-/Ausgabeport 3 - Der Ein-/Ausgabeport 3 dient, ebenso wie der Ein-/Ausgabeport 2 zur Eingabe und Ausgabe von acht Einzelsignalen, die allerdings fest aus einer 4-Bit-Eingabegruppe und einer 4-Bit-Ausgabegruppe bestehen (Bild 4.7.).

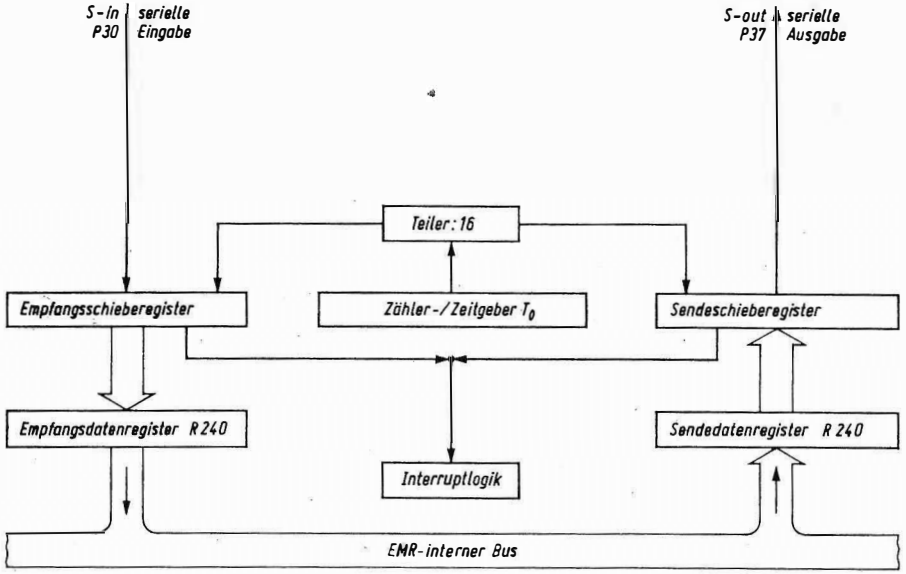


Bild 4.8. Interner Aufbau der seriellen Ein-/Ausgabe (UART) des U881/U882

Besonders zu beachten ist, daß der Ein-/Ausgabeport 3 bei Bedarf zur Vermittlung einer ganzen Reihe von speziellen Signalen verwendet werden kann:

Eingangsinterruptsignalleitungen (interrupt request)

P30	IRQ3
P31	IRQ2
P32	IRQ0
P33	IRQ1

Handshake-Quittungsbetriebssignale für die Steuerung des Ein-/Ausgabedatenverkehrs Port 0, 1 und 2

P31	DAV2 negiert / RDY2
P32	DAV0 negiert / RDY0
P33	DAV1 negiert / RDY1
P34	RDY1 / DAV1 negiert
P35	RDY0 / DAV0 negiert
P36	RDY2 / DAV2 negiert

serielle Signale für den integrierten duplex UART-Baustein zur bitseriellen Ein-/Ausgabe

P30 serielle Eingabe
P37 serielle Ausgabe

CTC-Ein-/Ausgabesignale 'Nulldurchgangsausgang' und 'Triggereingang'

P31 T-in
P36 T-out

DM-Signal (external data memory select) für den Anschluß externer Speicher

P34 DM negiert.

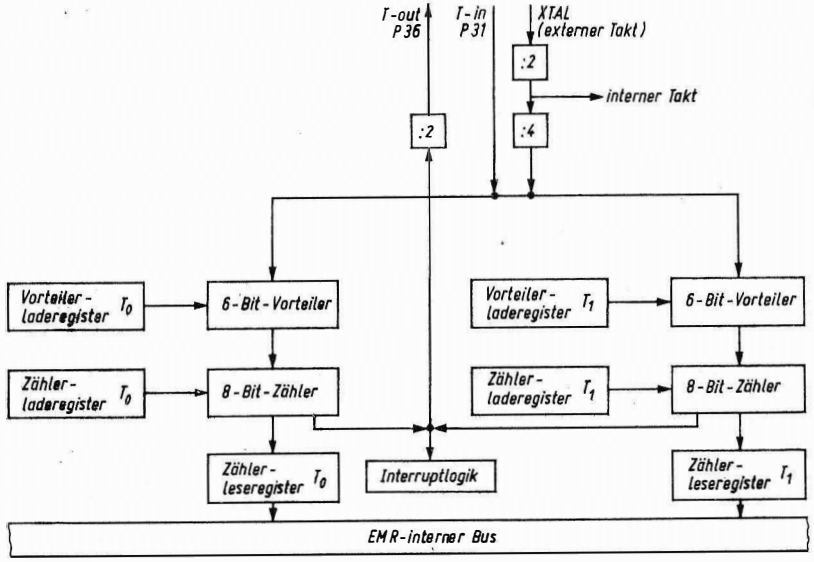


Bild 4.9. Interner Aufbau der Zähler-/Zeitgeberkanäle des U881/U882

Serielle Ein-/Ausgabe – Die bit-serielle UART-Ein-/Ausgabe mit dem U881/U882-EMR Baustein kann nur im Asynchronbetrieb mit oder ohne Paritätskontrolle mit folgenden Datentelegrammstrukturen erfolgen:

Sendedaten
1 Startbit - 8 Datenbits - 2 Stopbits
1 Startbit - 7 Datenbits - 1 Paritätsbit (odd) - 2 Stopbits

Empfangsdaten
1 Startbit - 8 Datenbits - 1 Stopbit
1 Startbit - 7 Datenbits - 1 Paritätsfehlerflag - 1 Stopbit

Die Datenbitrate für die serielle Ein-/Ausgabe wird über den CTC-Kanal T0 vorgegeben (wird intern noch dividiert durch 16 - d.h. Bitrate=XTAL:(2x4xPREOxTOx16)). Die serielle Dateneingabe und die serielle Datenausgabe selbst erfolgt vom Programm über Lese- und Schreibbefehle bezogen auf das Register R240 (Bild 4.8.).

Zähler-/Zeitgeber – Der CTC-Funktionsblock (CTC counter timer circuit) dient, je nach Initialprogrammierung, als Zähler oder als Zeitgeber, das heißt zum Zählen von externen Ereignissen über den Triggereingang oder zum Zählen des internen Systemtaktes. Er ist aus zwei adäquaten Kanälen aufgebaut, die jeweils aus einem 6-Bit-Vorteilerregister und aus einem 8-Bit-Zählregister bestehen. Die Betriebsart Zeitgeber dient insbesondere zum Aufbau einer EMR-internen Zeitbasis. Sie ist unabdingbare Voraussetzung für den Einsatz des Einchipmikrorechners in Echtzeitsystemen (Bild 4.9.).

4.2.2. Programmierung

Die Programmierung der Ein-/Ausgabekanäle des U881/U882-EMR erfolgt über die Register R240, R241 ... R248 (Tafel 4.2. und Bild 4.3.).

Tafel 4.2. Programmierung der Ein-/Ausgabeports des Einchipmikrorechners U881/U882

Für alle Register gilt der Aufbau:

D7	D6	D5	D4	D3	D2	D1	D0

Notation: Pxy Ein-/Ausgabebit y im Ein-/Ausgabeport x

Ein-/Ausgabeport 0 und 1 Moderegister R248(0F8H) P01M (Schreibregister)

D7,D6=00 P04...P07 Ausgabebits
D7,D6=01 P04...P07 Eingabebits
D7,D6=1x P04...P07 Adreßbits A12...A15 (x Bit ohne Bedeutung)

D5=0 normales Memory-Timing
D5=1 erweitertes Memory-Timing (eine zusätzliche Clockperiode)

D4,D3=00 P10...P17 Byteausgabe
D4,D3=01 P10...P17 Byteeingabe
D4,D3=10 P10...P17 Adreß-/Datenbits AD0...AD7 (multiplex)
D4,D3=11 P10...P17 hochohmig AD0...AD7, AS, DS, R/W, A8...A11, A12...A15 - wenn entsprechend programmiert

D2=0 externer Stack
D2=1 interner Stack

D1,D0=D0 P00...P03 Ausgabebits
D1,D0=01 P00...P03 Eingabebits
D1,D0=1x P00...P03 Adreßbits A8...A11 (x Bit ohne Bedeutung)

Ein-/Ausgabeport 2 Moderegister R246(0F6H) P2M (Schreibregister)

D7,D6...D0 Das entsprechende Bit des Port 2 wird als Ausgabebit (=0) oder Eingabebit (=1) definiert.

Ein-/Ausgabeport 3 Moderegister R247(0F7H) P3M (Schreibregister)

D7=0	Keine Paritätskontrolle bei der seriellen Datenübertragung			
D7=1	Paritätskontrolle bei der seriellen Datenübertragung			
D6=0	P30 Eingabebit	/	P37	Ausgabebit
D6=1	P30 serielle Eingabe	/	P37	serielle Ausgabe
D5=0	P31 Eingabebit (CTC T-in)	/	P36	Ausgabebit (CTC T-out)
D5=1	P31 Port 2 DAV2 negiert	/	P36	Port 2 ROY2
D4,D3=00	P33 Eingabebit	/	P34	Ausgabebit
D4,D3=01/10	P33 Eingabebit	/	P34	DM negiert
D4,D3=11	P33 Port 1 DAV1 negiert	/	P34	Port 1 RDY1
D2=0	P32 Eingabebit	/	P35	Ausgabebit
D2=1	P32 Port 0 DAV0 negiert	/	P35	Port 0 RDY0

D1=x nicht benutzt

D0=0 Port 2 Ausgänge (pull-ups: open drain)
D0=1 Port 2 Ausgänge (pull-ups: active)

Serielles Ein-/Ausgaberegister R240(0F0H) SIO (Lese-/Schreibregister)

D7,D6...D0 UART-Daten (D0 niederwertigstes Bit)
Lesen R240 - serielle Eingabe
Schreiben R240 - serielle Ausgabe

Zeitgebermoderegister R241(0F1H) TMR (Lese-/Schreibregister)

D7,D6=00 ohne Bedeutung (P36 ist ein Datenausgabebit / s. R247 D5)
D7,D6=01 T0-Ausgabe (T-out)
D7,D6=10 T1-Ausgabe (T-out)
D7,D6=11 Systemtaktausgabe (:2 T-out)

D5,D4=00 ext. Takteingang (T-in) / (ext. Takt:4 in Vorteiler)
D5,D4=01 ext. Toreingang (T-in) / (ext. Takt:4 in Vorteiler / ext. Startflanke 'high to low')
D5,D4=10 ext. Triggereingang (non-triggerable) (T-in) / (interner Takt:4 in Vorteiler / einmalige ext. Startflanke 'high to low')
D5,D4=11 ext. Triggereingang (retriggerable) (T-in) / (interner Takt:4 in Vorteiler / mehrmalige ext. Startflanke 'high to low')

D3=0 Ausschalten T1-Zählung
D3=1 Einschalten T1-Zählung

D2=0 nicht benutzt
D2=1 Laden von T1 (Übernahme R243 und R242)

D1=0 Ausschalten T0-Zählung
D1=1 Einschalten T0-Zählung

D0=0 nicht benutzt
D0=1 Laden von T0 (Übernahme R245 und R244)

T1-Zähler-/Zeitgeberregister R242(0F2H) T1 (Lese-/Schreibregister)

D7,D6...D0 Zählerinitialwert beim Schreiben (1...255,256 / 01...FF,00)
momentaner Zählerstand beim Lesen

T1-Vorteilerregister R243(0F3H) PRE1 (Schreibregister)

D7,D6,D5,D4,D3,D2= Vorteilerwert (1...63,64 / 01...FC,00)

D1=0 T1 externer Zähltakt (T-in Mode)
D1=1 T1 interner Zähltakt (interner Takt:4 in Vorteiler)

D0=0 Betriebsart Single-Pass (einmaliges Abzählen)
D0=1 Betriebsart Modulo-N (mehrmaliges Abzählen mit Autorestart)

T0-Zähler-/Zeitgeberregister R244(0F4H) T0 (Lese-/Schreibregister)

Die Bitbelegung des Zähler-/Zeitgeberregisters T0 (R244) entspricht der Bitbelegung des Zähler-/Zeitgeberregisters T1 (R242).

T0-Vorteilerregister R245(0F5H) PRE0 (Schreibregister)

Die Bitbelegung des Vorteilerregisters PRE0 (R245) entspricht der Bitbelegung des Vorteilerregisters PRE1 (R243).

8. U883 TINY-MPBASIC

Der Einchipmikrorechner U883 enthält in seinem 2 KByte großen internen ROM einen einfachen BASIC-Interpreter. Daneben ist ein dazugehöriger Editor-/Debuggerteil in Entwicklung, der alle Funktionen, die für die Programmentwicklung notwendig sind, enthält. Diese Komponenten liegen im externen ROM, sie können nach vollendeter Entwicklungsarbeit entfallen.
Damit wird vielen potentiellen Anwendern des Einchipmikrorechners, die über keine Entwicklungstechnik verfügen, eine Möglichkeit gegeben Programme zu entwickeln (z.B.: für Steuerungs- oder Regelungsaufgaben im Rationalisierungsmittelbau). Der Anwender kann somit seine Programme im Zusammenspiel mit der von ihm erstellten Hardware austesten. Nach der Erprobung ist das fertige Programm in einen EPROM zu laden, und das Gerät kann eingesetzt werden.
Da die Problemlösung in Form von BASIC-Programmen erarbeitet wird, ist ein schnelles Erstellen und Modifizieren der Anwendersoftware möglich, ohne daß ein großer Aufwand für die sonst notwendige Entwicklungstechnik auftritt.

8.1. Sprachkonzept und Anwendung

Das in TINY-MPBASIC geschriebene Anwenderprogramm wird vom im internen ROM-Bereich des U883 befindlichen BASIC-Interpreters abgearbeitet.
Neben dem BASIC-Programm sind Initialisierungsteile und, falls notwendig, die Prozeduren GET_CHAR (Einzelzeichenausgabe) und PUT_CHAR (Einzelzeichenausgabe) zu erstellen. Diese sehr stark vom Einsatzfall abhängigen Teile sind in Assemblersprache zu realisieren.
Nach durchgeführter Initialisierung kann das BASIC-Anwenderprogramm aufgerufen werden. Dies erfolgt durch einen CALL-Befehl zur Adresse %7FD. Vorher sind in Register 6 der höherwertige Teil und in Register 7 der niederwertige Teil der Startadresse zu laden. Der Anwender kann für seine Problemlösung zusätzlich noch externe Prozeduren und Funktionen in Assemblersprache realisieren, die vom BASIC aus aufrufbar sind. Das Vorhandensein externer Prozeduren und Funktionen wird dem Interpreter dadurch bekannt gemacht, daß in den Registern 8 und 9 die Adresse einer Prozedurnamentabelle übergeben wird. Falls keine externen Prozeduren verwendet werden, so sind die Register 8 und 9 vor Aufruf des BASIC-Interpreters Null zu setzen.
Das BASIC-Anwenderprogramm muß syntaktisch fehlerfrei sein. Es wird in verdichteter Form abgespeichert. Unter dem Betriebssystem UDOS existiert ein Konvertierungsprogramm (mit dem Namen COMPRIMIERE), das die verdichtete Form aus einer BASIC-Quelldatei herstellt. Dabei werden automatisch Standardprogrammteile zur Initialisierung und zur Einzelzeichenein- bzw. -ausgabe hinzugefügt.

Der TINY-MPBASIC-Interpreter verarbeitet intern 16 Bit breite Daten, die als Integergrößen (Zweierkomplementdarstellung: -32768 ... +32768), Wortgrößen oder Bytewerte (niederwertige 8 Bit) interpretiert werden können. Konstanten können in Dezimal- oder Hexadezimalschreibweise (durch das Prozentzeichen gekennzeichnet: %0 ... %FFFF) angegeben werden. Negative Dezimalzahlen müssen in Klammern gesetzt werden, damit das Vorzeichen nicht als Operator wirkt. Für Variablenbezeichnungen sind die Buchstaben A bis Z verwendbar. (Sie belegen im Registersatz die Adressen ab %30.)

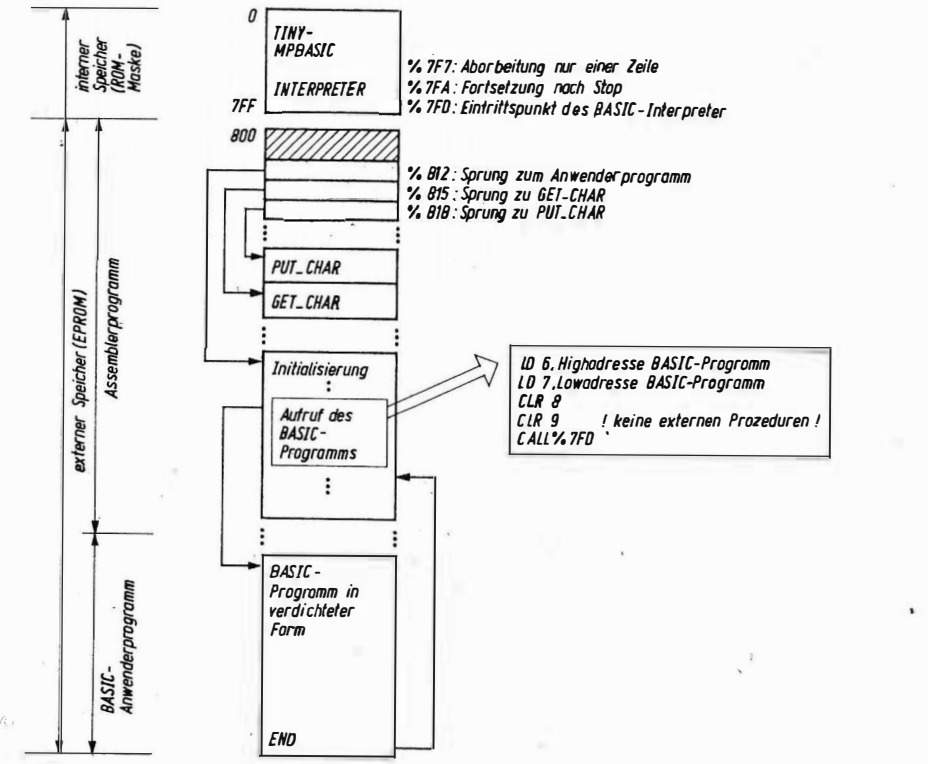


Bild 8.1. Zusammenspiel des U883-BASIC-Interpreters mit Anwenderteilen

Ausdrücke werden durch Verknüpfung von Konstanten, Variablen- oder Funktionswerten durch logische und arithmetische Operatoren gebildet.

Operatoren	
+	Addition
-	Subtraktion
*	Multiplikation
/	Division
$AND	logisches UND (bitweise)
$OR	logisches ODER "
$XOR	exklusives ODER "

Ausdrücke werden von links nach rechts ausgewertet. Es besteht die Möglichkeit der Klammerung. Die Verschachtelungstiefe hängt dabei von der verfügbaren Stackgröße ab.

Eine Prozedur ist ein in Assemblersprache geschriebenes Programm, das einen Satz von BASIC übergebenen Eingabeparametern verarbeitet und Ausgabeparameter an den BASIC-Interpreter zurückgibt. Die Parameterübergabe erfolgt im Stackbereich. Der Aufruf erfolgt über den Prozedurnamen.

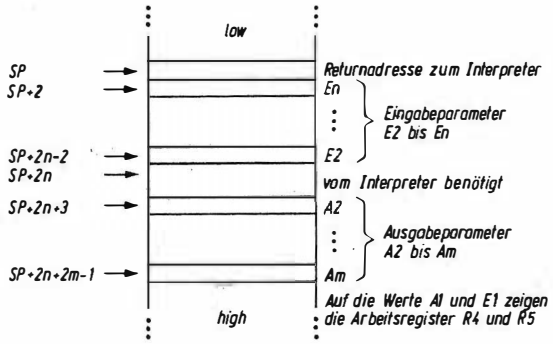


Bild 8.2. Parameterübergabeschema für externe Prozeduren

Funktionen sind Prozeduren, die genau einen Wert an den Interpreter übergeben. Sie können deshalb in Ausdrücken verwendet werden.
Neben einer Reihe von fest integrierten Standardprozeduren und -funktionen hat der Anwender die Möglichkeit eigene, seiner Hardwarekonfiguration angepaßte Prozeduren oder Funktionen, hinzuzufügen. Die Verbindung zu TINY-MPBASIC erfolgt über die schon erwähnte Prozedurnamentabelle.

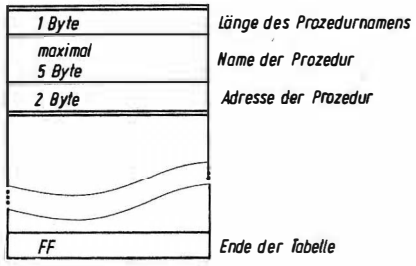


Bild 8.3. Aufbau der externen Prozedurnamentabelle

Das BASIC-Anwenderprogramm ist zeilenorientiert. Jede Zeile beginnt mit einer Nummer. Pro Zeile muß wenigstens eine Anweisung vorhanden sein. Mehrere Anweisungen auf einer Zeile sind möglich, wenn sie durch Semikolons getrennt sind. Die Programmzeilen müssen in aufsteigender Folge markiert sein. Die Anweisungsnamen werden in abgekürzter Form abgelegt, um möglicht wenig Speicherplatz zu belegen. Leerzeichen werden außer in Kommentaren und Texten entfernt. Die Sortierung und das Herstellen der verdichteten Form übernimmt normalerweise der Editorteil.

8.2. Anweisungen

Die Wertzuweisung für eine Variable ist die LET-Anweisung.

```
LET Variablenname = Ausdruck
```

Zur Programmverzweigung dient die GOTO-Anweisung.

```
GOTO Ausdruck
```

Unterprogramme können mit Hilfe der GOSUB-Anweisung aufgerufen werden. Das Ende eines Unterprogramms wird durch die RETURN-Anweisung markiert. Sie bewirkt die Programmfortsetzung bei der nach GOSUB folgenden Anweisung.

```
GOSUB Ausdruck
RETURN
```

Unterprogramme können weitere Unterprogramme aufrufen. Die Verschachtelungstiefe hängt vom verfügbaren Stackbereich ab.

Programmverzweigungen werden mit der IF/THEN-Anweisung realisiert. Sie hat folgende Form:

```
IF Ausdruck Vergleichsoperator Ausdruck THEN Anweisung
```

Für Vergleichsoperator kann dabei stehen:

=, <, >, <=, >= oder <> (für ungleich)

Falls die nach IF folgende Bedingung wahr ist, so wird die nach THEN folgende Anweisung ausgeführt. Andernfalls wird mit der nächstfolgenden Programmzeile fortgesetzt. Bei mehreren Anweisungen innerhalb einer Zeile wird der Rest der Programmzeile übersprungen!

Mit der PROC-Anweisung kann innerhalb des BASIC-Programms eine externe Prozedur aufgerufen werden.

```
PROC [Variablenliste] = Prozedurname [Parameterliste]
```

Die Variablen innerhalb der optionalen Variablenliste (Rückgabeparameter) und die Parameter innerhalb der ebenfalls optionalen Parameterliste (Eingabeparameter) sind durch Kommas zu trennen. Als Parameter sind Variablen und Konstanten zulässig.

Zur Ein-/Ausgabe dienen die Anweisungen PRINT, PRINTHEX und INPUT. Die Festlegung, über welche Geräte die Ein-/Ausgabeströme laufen, erfolgt durch die Programme GET_CHAR und PUT_CHAR. Dies kann je nach Anwendungsfall sehr unterschiedliche Hardware sein (z.B.: Terminal, Fernschreiber, LED- oder LCD-Anzeigen und verschiedene Tastaturen o.ä.).

```
PRINT ["beliebiger_Text"] [Ausdruck]
PRINTHEX ["beliebiger_Text"] [Ausdruck]
```

Die PRINT-Anweisungen geben einen Zahlenwert (bei PRINT: dezimal und bei PRINTHEX: hexadezimal) aus. Diesem Wert kann eine beliebige Textkette vorangestellt sein. Falls sowohl der Text als auch der Ausdruck fehlen, so wird lediglich ein Zeilenvorschub (%0D) ausgegeben.

```
INPUT ["beliebiger_Text"] Variablenname
```

Die INPUT-Anweisung gibt, falls vorhanden, erst den angegebenen Text aus und weist den eingegebenen Wert der spezifizierten Variable zu.

Zur Programmsteuerung dienen die STOP- und END-Anweisung.

```
STOP
END
```

Die Programmzeile, in der STOP auftritt, wird zu Ende abgearbeitet. Danach wird der Interpreter verlassen. Sie dient in Verbindung mit dem Editor/Debuggerteil zum Programmtest (Setzen von Unterbrechungspunkten). Durch Aufruf des Interpreters mit dem Eintrittspunkt %7FA kann der Programmablauf fortgesetzt werden.
Die END-Anweisung kennzeichnet das Programmende. Sie bewirkt ebenfalls das Verlassen des BASIC-Interpreters.

Zur Kommentierung der Programme dient die REM-Anweisung.

```
REM [Kommentartext]
```

Bei der Anwendung dieser Anweisung sollte der zur Verfügung stehende Gesamtspeicherplatz für das Anwenderprogramm bedacht werden.

Zur Erzeugung von Warteschleifen kann die WAIT-Anweisung verwendet werden.

```
WAIT Ausdruck
```

Sie bewirkt das Durchlaufen einer Softwarewarteschleife. Der Ausdruckswert spezifiziert dabei die Anzahl der Durchläufe. Beim Wert 1 wird die Schleife einmal und beim Wert Null 65536-mal durchlaufen. Bei einer Taktfrequenz von 4 MHz dauert ein Durchlauf eine Millisekunde.

Mit Hilfe der CALL-Anweisung kann ein in Assemblersprache geschriebenes Programm aufgerufen werden, ohne daß Parameter übermittelt werden.

```
CALL Ausdruck
```

Der Ausdruckswert ist die Programmadresse. Das Maschinenprogramm muß mit einem RET-Befehl enden.

Fest integriert im TINY-MPBASIC-Interpreter sind folgende Prozeduren:

NEG[Parameter]	arithmetische Negation
ABS[Parameter]	absoluter Betrag
NOT[Parameter]	logische Negation (bitweise)
GETR[Register]	liefert den Inhalt des angegebenen Registers (die höherwertigen 8 Bit sind Null)
GETRR[Register]	liefert den Inhalt des spezifizierten Registers (höherwertige 8 Bit) und des nachfolgenden Registers (niederwertige 8 Bit)
GETEB[Adresse]	holt Bytewert aus externem Speicher
GETEW[Adresse]	holt Wortwert aus externem Speicher

Analog können auch Register und Bytes (bzw. Wörter) im externen Datenspeicher gesetzt werden:

SETR[Register,Wert]	Register setzen
SETRR[Register,Wert]	Doppelregister setzen
SETEB[Adresse,Wert]	externes Byte setzen
SETEW[Adresse,Wert]	externes Wort setzen

Innerhalb der verdichteten Form des BASIC-Programms werden für die Anweisungen folgende Abkürzungen verwendet:

Anweisung	Abkürzung
LET	L
GOTO	G
IF...THEN	F...,
GOSUB	S
RETURN	R
PROC	O
INPUT	I
PRINT	P
PRINTHEX	H
STOP	T
END	E
REM	M
WAIT	W
CALL	C

Der restliche Programmtext wird ohne Leerzeichen in ASCII-Zeichen (die Zeilennummer als 2 Byte große Hexadezimalzahl) abgespeichert. Das Zeilenende wird durch %0D gekennzeichnet.

8.3. Programmbeispiel

Das folgende Demonstrationsbeispiel zur Anwendung von TINY-MPBASIC wurde aus /46/ übernommen. Das Bild 8.4. zeigt das zur Initialisierung und zum Programmstart notwendige Assemblerprogramm.
Die Bilder 8.5. und 8.6. zeigen ein BASIC-Demonstrationsprogramm in Quellform und in der verdichteten Form.

```
P 0812
P 0812 8D  081B
P 0815 8D  0841
P 0818 8D  0851
P 081B


P 081B

P 081B 8C  96
P 081D E6  FF  80
P 0820 E6  F4  40
P 0823 E6  F7  49
P 0826 E6  F5  0D
P 0829 E6  F1  43
P 082C B0  FB
P 082E 9F
P 082F E6  06  08
P 0832 E6  07  69
P 0835 B0  08
P 0837 B0  09
P 0839 B6  F0  23
P 083C D6  07FD
P 083F 8B  DA
P 0841


P 0841

P 0841 56  FA  F7
P 0844 76  FA  08
P 0847 6B  FB
P 0849 56  FA  F7
P 084C 88  F0
P 084E 56  E8  7F
P 0851

P 0851

P 0851 76  FA  10
P 0854 6B  FB
P 0856 56  FA  EF
P 0859 89  F0
P 085B A6  E8  0D
P 085E 6B  01
P 0860 AF
P 0861 8C  0A
P 0863 D6  0851
P 0866 8C  0D
P 0868 AF
P 0869


P 0869
```

```
1  U883TEST MODULE
2
3      ! MOEGLICHES ANWENDERPROGRAMM IM EXT. SPEICHER !
4  INTERNAL
5          EINTRITT PROCEDURE
6            ENTRY
7            ¤ABS %812      ! EINTRITTSPUNKTE !
8              JP BEGIN
9              JP GET_CHAR
10             JP PUT_CHAR
11         END EINTRITT
12
13     ! INITIALISIERUNG UND AUFRUF DES INTERPRETERS !
14
15         BEGIN PROCEDURE
16           ENTRY
17             LD R8,#%(2)10010110  ! NORMAL EXT. TIMING !
18             LD SPL,#%80
19             LD T0,#%40           ! TIMER FUER 300 BAUD  !
20             LD P3M,#%(2)01001001
21             LD PRE0,#%(2)00001101
22             LD TMR,#%(2)01000011
23             CLR IMR
24             EI
25             LD 6,#HI BASICPROGRAMM
26             LD 7,#LO BASICPROGRAMM
27             CLR 8                ! KEINE EXT. PROZEDUR !
28             CLR 9
29             LD SIO,#'#'          ! PROMPT-ZEICHEN  !
30             CALL %7FD            ! BASIC INTERPRETER !
31             JR BEGIN
32         END BEGIN
33
34     ! ZEICHEN HOLEN BZW. SENDEN
35       FERNSCHREIBER MIT FULL-DUPLEX !
36
37         GET_CHAR PROCEDURE
38           ENTRY
39             AND IRQ,#%F7       ! ALTEN REQUEST RUECKS. !
40 GTC10:       TM IRQ,#%8         ! ZEICHEN DA ?          !
41             JR Z,GTC10
42             AND IRQ,#%F7       ! REQUEST RUECKSETZEN   !
43             LD R8,SIO
44             AND R8,#%7F        ! PARITAET RUECKSETZEN  !
45         END GET_CHAR
46
47         PUT_CHAR PROCEDURE
48           ENTRY
49             TM IRQ,#%10        ! LETZTES ZEICHEN RAUS? !
50             JR Z,PUT_CHAR
51             AND IRQ,#%EF       ! REQUEST RUECKSETZEN   !
52             LD SIO,R8
53             CP R8,#'%r'
54             JR Z,PTC10
55             RET
56 PTC10:      LD R8,#'%l'        ! AUTO LINEFEED         !
57             CALL PUT_CHAR
58             LD R8,#'%r'
59             RET
60         END PUT_CHAR
61
62 ¤SECTION PROGRAM
63
64     ! BEGINN DES BASICPROGRAMMS !
65       BASICPROGRAMM ARRAY[0 BYTE]
66 END U883TEST
```

Bild 8.4. Assemblerprogramm für TINY-MPBASIC


```
  0     REM     BASICDEMONSTRATION
 10     PRINT "WAEHLEN SIE BITTE EIN PROGRAMMBEISPIEL !"
 20     PRINT
 30     PRINT "1 PRIMFAKTORZERLEGUNG"
 40     PRINT "2 UMRECHNUNG HEX-DEZIMAL"
 50     PRINT "3 UMRECHNUNG DEZIMAL-HEX"
 60     PRINT "4 REGISTERINHALT MODIFIZIEREN"
 70     PRINT "5 LANGSAM ALPHABET DRUCKEN"
 80     PRINT "6 NEU BEGINNEN"
 90     PRINT
100     INPUT "PROGRAMM NR ?: " A
110     GOTO 150*A
150     REM     PRIMFAKTORZERLEGUNG
160     INPUT "ZAHL=? " A
170     LET B = 2
180     IF A < 2, GOTO 240
190     LET C = A/B*B
200     IF C <> A, LET B = B+1; GOTO 190
220     PRINT B
230     LET A = A/B; GOTO 180
240     PRINT "FERTIG"
250     GOTO 100
300     REM     UMRECHNUNGEN
310     INPUT "HEXZAHL=? " A
320     PRINT "DEZIMAL = " A
330     GOTO 100
450     INPUT "DEZIMALZAHL=? " A
460     PRINTHEX "HEX = " A
470     GOTO 100
600     REM     REGISTERINH. MODIFIZ.
610     INPUT "REGISTER NR.: " A
620     PRINTHEX "INHALT = " GETR[A]
630     INPUT "NEUER INHALT: " B
640     PROC SETR[A,B]
650     GOTO 100
750     REM     ALPHABET DRUCKEN
760     INPUT "WARTEZEIT ZWISCHEN ZWEI BUCHSTABEN [MSEC]:" A
770     LET B = 26; LET C = %41
780     LET Z = C; GOSUB 1000
790     LET Z = %20; GOSUB 1000; REM SPACE DAZWISCHEN
800     WAIT A
810     LET C = C+1; LET B = B-1
820     IF B <> 0, GOTO 780
830     LET Z = %D; GOSUB 1000; REM CR & LF ANHAENGEN
840     LET Z = %A; GOSUB 1000
850     GOTO 100
900     REM     PROGRAMM VERLASSEN
910     END
1000    IF GETR[%FA] ¤A %10 <> %10, GOTO 1000
1010    PROC SETR[%FA,0]; REM REQUEST RUECKSETZEN
1020    PROC SETR[%F0,Z]
1030    RETURN
2000    ENDE
```

Bild 8.5. TINY-MPBASIC-Programm (Quellform)

```
FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF *........:.Q.....*
FF FF 8D 08 1B 8D 08 41 8D 08 51 8C 96 E6 FF 80 *.......A..Q.....*
E6 F4 40 E6 F7 41 E6 F5 0D E6 F1 43 B0 FB 9F E6 *..@..A.....C....*
06 08 E6 07 69 B0 08 B0 09 E6 F0 53 D6 07 FD 8B *....i......S....*
DA 56 FA F7 76 FA 08 6B FB 56 FA 76 88 F0 56 E8 *.V..v..k.V.v...V.*
7E 76 FA 10 6B FB 56 FA EF 89 F0 A6 E8 0D 6B 01 *.v..k.V.......k.*
AF 8C 0A D6 08 51 8C 0D AF 00 00 4D 42 41 53 49 *.....Q....MBASI*
AF 8C 0A D6 08 51 8C 0D AF 00 00 4D 4F 4E 0D 00 *.....Q....MBASI*
43 44 45 4D 4F 4E 53 54 52 41 54 49 4F 4E 0D 00 *CDEMONSTRATION..*

0A 50 22 57 41 45 48 4C 45 4E 20 53 49 45 20 42 *.P"WAEHLEN SIE B*
49 54 54 45 20 45 49 4E 20 50 52 4F 47 52 41 4D *ITTE EIN PROGRAM*
4D 42 45 49 53 50 49 45 4C 20 21 22 0D 00 14 50 *MBEISPIEL !"...P*
0D 00 1E 50 22 50 52 49 4D 46 41 4B 54 4F 22 50 *...P"PRIMFAKTO"P*
52 5A 45 52 4C 45 47 55 4E 47 20 48 45 58 22 50 *RZERLEGUNG"..("P*
32 20 55 4D 52 45 43 48 4E 55 4E 47 20 48 45 58 *2 UMRECHNUNG HEX*
2D 44 45 5A 49 4D 41 4C 22 0D 00 32 50 22 33 20 *-DEZIMAL"..2P"3 *
55 4D 52 45 43 48 4E 55 4E 47 20 44 45 5A 49 4D *UMRECHNUNG DEZIM*

41 4C 2D 48 45 58 22 0D 00 3C 50 22 34 20 52 45 *AL-HEX"..<P"4 RE*
47 49 53 54 45 52 49 4E 48 41 4C 54 20 4D 4F 44 *GISTERINHALT MOD*
49 46 49 5A 49 45 52 45 4E 22 0D 00 46 50 22 35 *IFIZIEREN"..FP"5*
20 4C 41 4E 47 53 41 4D 20 41 4C 50 48 41 42 45 * LANGSAM ALPHABE*
54 20 44 52 55 43 4B 45 4E 22 0D 00 50 50 22 36 *T DRUCKEN"..PP"6*
20 4E 45 55 20 42 45 47 49 4E 4E 45 4E 22 0D 00 * NEU BEGINNEN"..*
5A 50 0D 00 64 49 22 50 52 4F 47 52 41 4D 20 20 *ZP..dI"PROGRAMM  *
4E 52 20 3F 3A 20 22 41 0D 00 6E 47 31 35 30 2A *NR ?: "A..nG150**

41 0D 00 96 4D 50 52 49 4D 46 41 4B 54 4F 52 5A *A...MPRIMFAKTORZ*
45 52 4C 45 47 55 4E 47 0D 00 A0 49 22 5A 41 48 *ERLEGUNG...I"ZAH*
4C 3D 3F 20 22 41 0D 00 AA 4C 42 3D 32 0D 00 B4 *L=? "A...LB=2...*
46 41 3C 32 20 47 32 34 30 0D BE 4C 43 3D 41 *FA<2,G240...LC=A*
2F 42 2A 42 0D 00 C8 46 43 3C 3E 41 2C 4C 42 3D */B*B...FC<>A,LB=*
42 2B 31 3B 47 31 39 30 0D DC 50 42 0D 00 E6 *B+1;G100...PB...*
4C 41 3D 41 2F 42 3B 47 31 38 30 0D 00 F0 50 22 *LA=A/B;G180...P"*
46 45 52 54 49 47 22 0D 00 FA 47 31 30 30 0D 01 *FERTIG"...G100..*

2C 4D 55 4D 52 45 43 48 4E 55 4E 47 45 4E 0D 01 *,MUMRECHNUNGEN..*
36 49 22 48 45 58 5A 41 48 4C 3D 3F 20 22 41 0D *6I"HEXZAHL=? "A.*
01 40 50 22 41 45 5A 49 4D 41 4C 20 3D 20 22 41 *.@P"DEZIMAL = "A*
0D 01 4A 47 31 30 30 0D 3F 20 22 41 0D 01 CC 48 *..JG100..? "A...H*
4D 41 4C 5A 41 48 4C 3D 41 0D 01 D6 47 31 30 30 *MALZAHL=? "A...G100*
22 48 45 58 3D 20 22 41 0D 01 E0 49 4E 48 2E *"HEX = "A...G100*
0D 02 58 4D 52 45 47 49 53 54 45 52 49 4E 48 2E *..XMREGISTERINH.*
20 4D 4F 44 49 46 49 5A 2E 0D 02 62 49 22 52 45 * MODIFIZ...bI"RE*

47 49 53 54 45 52 20 4E 52 2E 3A 20 22 41 0D 02 *GISTER NR.: "A...*
4E 48 41 4C 54 20 3D 20 22 47 45 54 0D 02 86 49 *INHALT = "GET...I*
52 41 5D 20 22 4E 45 55 45 52 45 20 49 0D 02 98 *R[A] "NEUER I...*
42 41 4C 54 3A 20 22 42 0D 02 80 4F 53 45 54 *NBALT: "B...OSET*
52 5B 41 5D 20 47 31 30 30 0D 02 EE *R[A] G100...*
4D 41 4C 50 48 41 42 45 54 20 44 52 55 43 4B 45 *MALPHABET DRUCKE*
41 4C 50 48 41 42 45 54 20 44 52 55 43 4B 45 *ALPHABET DRUCKE*
4E 0D 02 F8 49 22 57 41 52 54 45 4E 20 42 55 *N...I"WARTEZEIT  *
5A 57 49 53 43 48 45 4E 20 5A 57 45 49 20 42 55 *ZWISCHEN ZWEI BU*

43 48 53 54 41 42 45 4E 20 5B 4D 53 5D 3A *CHSTABEN [MSEC];*
22 41 0D 03 02 4C 42 3D 32 36 3B 4C 43 3D 35 34 *"A...LB=26;LC=%4*
31 0D 03 0C 4C 5A 3D 43 3B 53 31 30 30 30 3B 4D 53 *1...LZ=C;S1000;MS*
16 4C 5A 3D 25 32 30 3B 53 31 30 30 30 3B 4D 53 *.LZ=%20;S1000;MS*
50 41 43 45 20 44 41 5A 57 49 53 43 48 45 4E 0D *PACE DAZWISCHEN.*
03 20 57 41 0D 03 2A 4C 57 49 53 43 48 45 4E 0D *. WA..*LC=C+1...*
3D 42 2D 31 0D 03 34 46 43 3D 43 2B 31 3B 4C 42 *=B-1..4FB<>0,G78*
30 0D 03 3E 4C 5A 3D 25 44 3B 53 31 30 30 30 3B *0..>LZ=%D;S1000;*

4D 43 52 20 26 20 4C 46 20 41 4E 48 41 45 4E 47 *MCR & LF ANHAENG*
45 4E 0D 03 48 4C 5A 3D 25 47 3B 53 31 30 30 30 *EN..HLZ=%A;S1000*
0D 03 52 47 31 30 30 0D 03 84 4D 50 52 4F 47 52 *..RG100...MPROGR*
41 4D 4D 20 56 45 52 4C 41 53 53 45 4E 0D 03 8E *AMM VERLASSEN...*
45 0D 03 E8 46 47 45 54 52 5B 25 46 41 5D 24 41 *E...FGETR[%FA]$A*
25 31 30 3C 3E 25 31 30 2C 47 31 30 30 30 0D 03 *%10<>%10,G1000..*
F2 4F 53 45 54 52 5B 25 46 41 2C 30 5D 3B 4D 22 *.OSETR[%FA,0];MR*
45 51 55 45 53 54 20 52 55 45 43 4B 53 45 54 5A *EQUEST RUECKSETZ*

45 4E 0D 03 FC 4F 53 45 54 52 5B 25 46 30 2C 5A *EN...OSETR[%F0,Z*
5D 0D 04 06 52 0D 07 D0 45 0D FF FF FF FF FF FF *]...R...E.......*
FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF *................*
FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF *................*
FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF *................*
FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF *................*
FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF *................*
FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF *................*
```

Bild 8.6. TINY-MPBASIC-Programm (verdichtete Form)

9. U880 PASCAL-Interpreter

Die Programmiersprache PASCAL hat sich in den letzten Jahren einen entscheidenden Platz bei der Aus- und Weiterbildung, der System- und Anwenderprogrammierung erobert. Der Grund für diesen Erfolg liegt sowohl in der klaren, logischen Struktur von PASCAL als auch in der guten Übertragbarkeit auf verschiedene Rechnertypen (Portabilität). Das Spektrum der Implementationen reicht von modernen Großrechnern bis zu Mikrorechnern.

Unter dem Betriebssystem UDOS existiert für den U880 ein leistungsfähiges dialogorientiertes PASCAL-System, das für ein breites Anwendungsspektrum (von kommerziellen Projekten bis zu numerischen Berechnungen) genutzt werden kann.

In diesem Abschnitt werden die notwendigen Informationen zur Übersetzung und Abarbeitung von PASCAL-Quellprogrammen unter dem Betriebssystem UDOS dargelegt. Ebenso werden die Einschränkungen und Erweiterungen dieser PASCAL-Version erläutert und Hinweise zur Handhabung gegeben.

Für eine ausführliche PASCAL-Sprachbeschreibung sei auf die Standardliteratur /49/, /50/, /51/ u.a. verwiesen.

Das PASCAL-System benutzt einige Möglichkeiten des Betriebssystems UDOS, wie reihenweise Ein-/Ausgabe zur Dateiarbeit und die Speicherverwaltung (memory manager). Eine umfassende Beschreibung dieser Möglichkeiten ist der Anwenderdokumentation zum Betriebssystem /43/ zu entnehmen.

9.1. Arbeitsweise

Das PASCAL-System besteht aus drei Programmen: einem PASCAL-Compiler, einem Post-Prozessor und einem Interpreter. Der Compiler, selbst ein PASCAL-Programm, erzeugt P-Kode, einen symbolischen Zwischenkode für einen stack-orientierten Rechner. Der Post-Prozessor konvertiert den vom Compiler erzeugten P-Kode zur Abarbeitung auf kleineren Rechnern in einen verdichteten Objektkode, der dann unter Steuerung des Interpreters abgearbeitet wird.

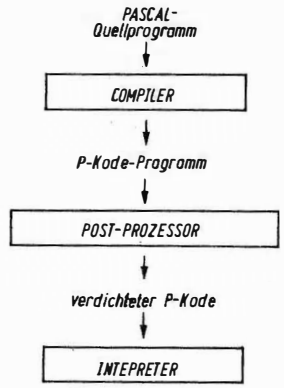


Bild 9.1. PASCAL-Ablaufübersicht